Algorithmic Operations Research

Preemptive scheduling with position costs

Francis Sourd

Volume 1, Number 2, Summer 2006

URI: https://id.erudit.org/iderudit/aor1_2art03

See table of contents

Publisher(s)

Preeminent Academic Facets Inc.

ISSN

1718-3235 (digital)

Explore this journal

Cite this article

Sourd, F. (2006). Preemptive scheduling with position costs. *Algorithmic Operations Research*, *1*(2), 79–93.

Article abstract

This paper is devoted to basic scheduling problems in which the scheduling cost of a job is not a function of its completion time. Instead, the cost is derived from the integration of a cost function over the time intervals on which the job is processed. This criterion is specially meaningful when job preemption is allowed. Polynomial algorithms are presented to solve some special cases including a one-machine problem with a common due date and a two-machine problem with linear nondecreasing cost functions.

Algorithmic Operations Research Vol.1 (2006) 79–93

Preemptive scheduling with position costs

Francis Sourd [a]

[a]LIP6 - CNRS - Université Paris 6, 4, place Jussieu 75252 PARIS CEDEX 05

Abstract

This paper is devoted to basic scheduling problems in which the scheduling cost of a job is not a function of its completion time. Instead, the cost is derived from the integration of a cost function over the time intervals on which the job is processed. This criterion is specially meaningful when job preemption is allowed. Polynomial algorithms are presented to solve some special cases including a one-machine problem with a common due date and a two-machine problem with linear nondecreasing cost functions.

Key words: Scheduling algorithm, preemption, primal-dual algorithm, dynamic programming.

1. Introduction

In most scheduling models presented in the literature [5,14], the cost for scheduling a job J_i is a function of its completion time, usually denoted by C_i. When preemption is not allowed, the job must be wholly scheduled in the time interval $[C_i - p_i, C_i)$ where p_i denotes the processing time of J_i. Hence, the processing period of J_i is unambiguously defined by C_i. When preemption is allowed, there may be an infinite number of ways to schedule J_i so that it completes at C_i. Therefore, a model in which the cost of J_i only depends on C_i might be insufficient especially when the material produced by the execution of J_i is continuously delivered to the consumer, that is a fraction of the material is delivered as soon as it is produced instead of being entirely delivered at its completion time.

However, it does not mean that the problem was ignored by practitioners and researchers. In fact, it is usually considered at the planning level. The planning horizon is divided into time periods. In these models, the whole production is not processed in a single period, and production and holding costs are introduced in order to penalize a part of the production that would be processed in a bad time period with respect to the demand. More details about lot-sizing problems can be found in recent surveys [15,3]. The notion of preemption is also central in problems of balancing production lines: the production of different products must be rotated to satisfy the different types of demands while limiting inventories and shortages. A survey of these problems can be found in [12].

In this paper, we consider a new model which corresponds to a new criterion in the classic machine scheduling theory [5]. As detailed in Section 2., the main interest of the model is to avoid the use of time periods (the number of time periods is usually not polynomial). A cost function f_i is attached to each job J_i. It can be interpreted as follows: for an infinitesimal positive duration $\mathrm{d}t$, $f_i(t)\mathrm{d}t$ is the cost for processing J_i between t and $t + \mathrm{d}t$ (mathematical assumptions about f_i are given later). Therefore, the total cost of job J_i —called *position cost* of J_i— is $\int_0^\infty f_i(t)x_i(t)\mathrm{d}t$, where x_i is a 0-1 function that indicates whether J_i is processed or not at each time. For obvious practical reasons, x_i should be constrained to have a finite number of discontinuities, which means that the number of interruptions of each job J_i is finite. We will prove, for all the models presented in this paper, that the proposed algorithms compute optimal solutions which satisfy this condition. Finally, we observe that the condition that J_i is completely processed is expressed by the equation $\int_0^\infty x_i(t)\mathrm{d}t = p_i$. As a trivial consequence, we have that if the cost functions are modified by an additional constant, that is the cost functions are $f_i(t) + \lambda_i$, the change of the cost of a feasible schedule is $\sum_i \lambda_i p_i$, which is a constant. Therefore, an optimal schedule is invariant with respect to the choice of the λ_i.

The optimization criterion for the problems addressed in this paper is the minimization of the sum of the position costs of all the jobs, which will be denoted by $\sum\int f_i$ in the γ-field of the usual $\alpha \,|\, \beta \,|\, \gamma$ notation. Alternatively, the max criterion $\max \int f_i$ could also be of

Email: Francis Sourd [Francis.Sourd@lip6.fr].

interest but is not studied here. Moreover, the jobs are assumed to be independent, that is there is no precedence constraint between any pair of jobs.

Preemptive scheduling in order to minimize the total position costs also stems from the need of lower bounds for non-preemptive scheduling problems. An early approach was proposed by Gelders and Kleindorfer [9] for the single machine weighted tardiness problem. An extension of this approach has been proposed by Clifford and Posner [7] for the earliness-tardiness problem with a common due date and by Sourd and Kedad-Sidhoum [18] and Bülbül et al. [6] for the problem with general due dates. These lower bounds are shown to be efficient and can be used in a branch-and-bound method [18] and in lower-bound based heuristics [18,6]. In these papers, the lower bounds are not strictly defined as problems with position costs, instead the jobs are decomposed into unary operations, each operation being given a cost function. Sourd [16] introduces a position cost based model to compute a lower bound for the earliness-tardiness problem and presents the relationships between the new model and the ones of [18] and [6].

Section 2. provides some generalities and related results. It is explained that no efficient strongly polynomial algorithms for problems with the $\sum\int f_i$ criterion have been proposed in the literature. Therefore, the aim of the paper is to study classes of problems which can be well solved by combinatorial algorithms. Section 3. is then devoted to two special cases of the one-machine problem. In Section 4., a parallel machine problem with two machines is studied. Finally, Section 5 underlines the originality of the techniques used to solve these problems with position costs and indicates some open problems.

2. Generalities and motivation

Let us first consider the more general problem $P\,|\,|\,\sum\int f_i$. We first observe that this problem is equivalent to the problem with release dates and deadlines $P\,|\,r_i,\bar{d}_i\,|\,\sum\int f_i$ because the value of f_i can be set to an arbitrarily large value outside the time interval $[r_i,\bar{d}_i)$. Classically, the number of jobs is denoted by n and, for each i in $\{1,\ldots,n\}$, p_i denotes the processing time of J_i.

We also introduce some assumptions about the cost functions f_i and x_i in order to avoid technical issues. First, in order that the integral $\int_0^\infty f_i(t)x_i(t)\mathrm{d}t$ always exists, we assume that all the f_i are piecewise continuous. Second, if $f_i(t)$ is nonincreasing when t becomes infinite, then J_i should be scheduled at an infinitely late date. Therefore, f_i is assumed to be nondecreasing on a time interval $[A_i,\infty)$ where $A_i \geq 0$ is defined with respect to f_i. It can then be observed that there is an optimal schedule in which no job is processed after the *horizon* $T = \max_i A_i + \sum_i p_i$. This assumption on the cost functions f_i implies that there is an optimal solution in which the number of interruptions is finite [16]. Therefore, we will limit our attention to these schedules, which also means that we use the Riemann integral.

Let us now assume that all the job starts and interruptions must occur at integer time points. Then, a job J_i running or starting at t is processed within the whole interval $[t,t+1)$ and the contribution of this part of J_i to the total cost is equal to $\int_t^{t+1} f_i(t)\mathrm{d}t$. The problem is classicaly solved as a *transportation* problem (see for instance [1,5]).

As the processing times of the jobs are generally greater than 1, the resulting network is generally *unbalanced*, which means that the number of demands (sinks) is significantly greater than the number of suppliers (sources). Hence, the problem can be efficiently solved by some variants of the classical network flow algorithms for unbalanced bipartite networks (Ahuja and al. [2]). Sourd and Kedad-Sidhoum [18] also presented a variant of the Hungarian algorithm that solves the problem in the special case $m = 1$.

The main advantage of solving the scheduling problem as a transportation problem is the generality of the approach: it works as soon as $\int_t^{t+1} f_i$ can be computed —or approximated by a numerical method— for any t. However, when f_i can be compactly encoded, the approach is not so efficient. For instance, let us consider the case where we simply have $f_i(t) = |t - d_i|$, that is the function is encoded in $O(\log d_i)$ space. Then, the horizon T is not polynomial in the size of the input of the problem even if it is bounded by a polynomial in the values $p_1,\ldots,p_n$ and $d_1,\ldots,d_n$. Therefore, the transportation-based algorithm is not polynomial but pseudo-polynomial. We can also interpret this problem as the problem of scheduling $\sum_{i=1}^n p_i$ unit operations.

As the p_i unit operations derived from J_i are identical, the problem is related to the field of high multiplicity scheduling (see [4] for a recent discussion of these problems). In particular, Clifford and Posner [7] study several common due date earliness-tardiness scheduling problems in the context of high multiplicity. They propose polynomial algorithms which relies on the solving of several linear programs. In particular, they show that

when the processing times of the jobs are equal to 1, the problem is polynomial. This problem is similar to the problem we study in Section 3.2. when we add the constraint that interruptions must happen at integer time points. In contrast, we show that in our model with position costs, the algorithm is simpler and more efficient.

In what concerns position cost as defined in this paper, Sourd [16] studies the one-machine scheduling problem when the cost functions f_i are piecewise linear (job interruptions are no more forced to occur at integer times). It can be shown that the dual of this problem is the maximization of a non-smooth concave function and, as there is no duality gap, the problem is polynomial. When there are parallel machines instead of a single machine, the problem can be similarly solved (Kedad-Sidhoum et al. [11]). However, in a computational view, obtaining the optimal schedule, even if polynomial, is not so easy because the optimum often corresponds to a non-smooth point. Furthermore, the algorithm is not combinatorial and not strongly polynomial.

This observation motivates the study presented in this paper. Some special cases of the single and parallel machine problems are considered and polynomial combinatorial algorithms to solve them are given.

3. One-machine problems

3.1. *Release dates and linear cost functions*

We first consider that the cost functions are of the form

$$f_i(t) = \begin{cases} \infty & \text{if } t < r_i \\ w_i t & \text{otherwise} \end{cases}$$

where $r_i \geq 0$ is the *release date* of J_i and $w_i > 0$ is the *slope* of its cost function. We show that an optimal schedule can be obtained by scheduling the job according to the preemptive "largest slope first" rule, which is more formally described by Algorithm 1.

The algorithm can be proved by a classic interchange argument. As mentioned in Section 2., we limit our attention to schedules with a finite number of preemptions. Let us consider a schedule such that there is an interval $[t_1, t_1 + \delta_1)$ in which the scheduled job (say J_i) violates the "largest slope first" rule. As the rule is violated, there is a time interval $[t_2, t_2 + \delta_2)$ with $t_2 > t_1$ in which the scheduled job J_j verifies $w_j > w_i$ and $r_j \leq t_1$. Let us define $\delta = \min(\delta_1, \delta_2)$. The part of J_i scheduled in $[t_1, t_1 + \delta)$ can the be swapped with the part of J_j in $[t_2, t_2 + \delta)$. Simple calculations show that the change of the cost of the schedule is $(w_i - w_j)(t_2 - t_1) < 0$, which proves that the initial schedule was not optimal.

Algorithm 1 Preemptive "largest slope first" rule

let $t = \min_i r_i$
repeat
  select $J_{i^\star}$ with $r_i \geq t$ and $p_i > 0$ with the minimal slope w_i
  let $t' = \min\left(t + p_{i^\star}, \min\{r_i \mid r_i > t\}\right)$
  schedule $J_{i^\star}$ between t and t'
  decrease $p_{i^\star}$ by $t' - t$
  let $t = \max\left(t', \min\{r_i, p_i > 0\}\right)$
until $p_i = 0$ for all jobs

The algorithm can clearly be implemented in $O(n \log n)$ time. Since the processing of a job can only be interrupted by the release of another job, there are at most $n - 1$ interruptions in the schedule. In particular, there is no job interruption when all the jobs are simultaneously released.

3.2. *Earliness-tardiness around a common due date*

In this section, we consider that the cost functions are of the form $f_i(t) = \alpha_i \max(d - t, 0) + \beta_i \max(t - d, 0)$ for a *common* due date $d > 0$. For each job J_i, we also have $\alpha_i \geq 0$ and $\beta_i > 0$ because a job with $\beta_i = 0$ can be scheduled after all the other jobs and there is no release dates. A part of a job scheduled before d is said to be *early* otherwise it is *late*. We assume without loss of generality that $\alpha_i \neq \alpha_j$ and $\beta_i \neq \beta_j$ for any $i \neq j$. This assumption is based on [16, Remark 3] that shows that we sligthly modify the value of the slopes with a very small perturbation the optimal schedule is very slightly modified. Moreover, while this assumption makes the proof shorter, the algorithm we propose can be easily implemented to deal with equal slopes without explicitly introducing the small perturbations.

This problem is clearly related to the class of scheduling problems with earliness and tardiness penalties and a common due date. Two recent surveys of these problems have been proposed by Gordon et al. [10] and by Lauff and Werner [13] but only non-preemptive scheduling is addressed in both articles. As mentionned in Section 2., the closest model is the one proposed by Clifford and Posner [7] that considers unit jobs, which is similar to constrain interruption times to be integer.

We first observe that there is an optimal solution without idle time that completes at some time t such that

$t - P \leq d \leq t$ with $P = \sum_{i=1}^{n} p_i$. For a given schedule and for any i, let p_i^+ denote the length of J_i scheduled after d and let $p_i^- = p_i - p_i^+$ be the length of J_i which is early. Clearly, according to Section 3.1., in an optimal schedule, the late part of any job is not interrupted and the parts of jobs are sequenced in the order of nonincreasing β_i-costs. Similarly, the early parts are sequenced in the order of nondecreasing α_i-costs. Therefore, these dominant schedules are mathematically defined by the vector $p^+ = (p_1^+, \ldots, p_n^+)$: from this vector, we can easily derive the early parts of the jobs and the start times of both early and tardy parts. Let us define the *pseudo start time* of J_i $t_i^- = d - \sum_{\alpha_j \geq \alpha_i} p_j^-$ and its *pseudo completion time* $t_i^+ = d + \sum_{\beta_j \geq \beta_i} p_j^+$. If $p_i^+ = 0$, J_i is *wholly early* and is scheduled in the interval $[t_i^-, t_i^- + p_i)$ otherwise it is —at least partially— late and it completes at t_i^+. If $p_i^+ = p_i$, J_i is *wholly late* and is scheduled in the interval $[t_i^+ - p_i, t_i^+)$ otherwise it is —at least partially— early and it starts at t_i^-. The cost of the schedule is then denoted by $\mathrm{COST}(p^+)$.

In order to solve the problem [1], we define the parameterized problem $\mathcal{P}(t)$, which is the variant of our problem in which the jobs are forced to be scheduled in the time interval $[t - P, t)$:

$$\mathcal{P}(t): \ \min \sum_{i=1}^{n} \int_{t}^{t+P} f_i(\theta) x_i^t(\theta) \mathrm{d}\theta \quad (1)$$

$$\text{s.t.} \sum_{i=1}^{n} x_i^t(\theta) \leq 1 \forall \theta \in [t, t+P) \quad (2)$$

$$\int_{t}^{t+P} x_i^t(\theta) \mathrm{d}\theta = p_i \forall i \in \{1, \ldots, n\} \quad (3)$$

$$x_i^t(\theta) \in \{0,1\} \forall \theta \in [t, t+P) \quad \forall i \in \{1, \ldots, n\} \quad (4)$$

For any $t \in [d, d+P]$, we will consider the optimal solution x^t of $\mathcal{P}(t)$ that also verifies the above dominance properties. This solution is described by the vector $p^+(t) = \left(p_1^+(t), \ldots, p_n^+(t)\right)$ where $p_i^+(t) = \int_d^{t+P} x_i^t(\theta) \mathrm{d}\theta$.

Clearly, if $t = d + P$, all the jobs are wholly late so that they are sequenced in the order of the nonincreasing β_i. Conversely, if $P \leq t \leq d$ (if possible), all the jobs are wholly early and they are sequenced in the order of their nondecreasing α_i-values. Therefore, we study the problem when t varies in the interval $[\max(d, P), d+P]$. The aim of the proposed algorithm is to enumerate the optimal schedules when t varies in this interval in order to find the optimal cost of $\mathcal{P}(t)$, denoted by $\mathrm{OPT}(t)$, and given by

In our approach, we are going to solve the dual problem of (1-4) which consists in the unconstrained maximization of

$$q_t(\mu) = \sum_{i=1}^{n} \mu_i p_i + \int_{t}^{t+P} \min_{1 \leq i \leq n} (f_i(\theta) - \mu_i) \mathrm{d}\theta \quad (5)$$

for $\mu \in \mathbb{R}^n$. The optimal solution of this dual problem is denoted by $\mu(t)$. Then, the optimal cost $\mathrm{OPT}(t)$, is equal to $\mathrm{COST}(p^+(t)) = q_t(\mu(t))$. For a given value of μ, we can build a dual pseudo-schedule that execute, for each time $\theta \in [t-P, t]$, the job J_i that minimizes the value $f_i(\theta) - \mu_i)$. In general, this pseudo-schedule is not feasible because the time spent to process a job differs from the required processing time but it is feasible (and optimal) when $\mu = \mu(t)$. The relationship between the primal and dual solutions is central in the rest of the section. The reader can refer to [16] for more details.

We now present the main theorem of the section that shows how $\mathcal{P}(t-\epsilon)$ can be efficiently solved when the optimal solution of $\mathcal{P}(t)$ has already been computed. It is illustrated by Figure 1. In the proof of the theorem and in the following, $\mathcal{P}(t)$ is considered as the "current" problem so that the reference to t will be omitted in notations, that is p^+ and μ will denote $p^+(t)$ and $\mu(t)$.

Theorem 1 *For any $t > \max(P, d)$, there exists a value $\delta > 0$ and a job index $i^\star \in \{1, \ldots, n\}$ such that, for any $\epsilon \in [0, \delta]$,*

$$p_i^+(t-\epsilon) = \begin{cases} p_i^+(t) & \text{if } i \neq i^\star \\ p_{i^\star}^+(t) - \epsilon & \text{if } i = i^\star \end{cases}$$

In other words, an optimal solution of $\mathcal{P}(t-\epsilon)$ is derived from the solution of $\mathcal{P}(t)$ by making early the tardy quantity ϵ of some job $J_{i^\star}$, which is called the transfered job.

PROOF. In this proof, we consider the pair (p^+, μ) formed by the primal and dual solutions of $\mathcal{P}(t)$ and we build the optimal pair $(p^+(t-\epsilon), \mu(t-\epsilon))$. From the dual solution μ, let us define the sets

[1] An animated demonstration of the algorithm can be downloaded from the site of the author *http://www-poleia.lip6.fr/∼sourd/project/position*.

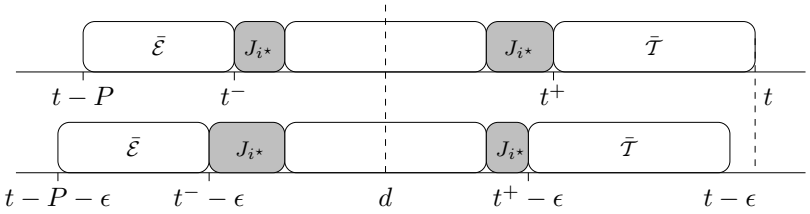

Fig. 1. From $\mathcal{P}(t)$ to $\mathcal{P}(t-\epsilon)$

$$\begin{aligned}\mathcal{E} = \mathcal{E}(t) = \{J_i \,|\, \exists \theta \in [d,t],\ \ \forall j, f_i(\theta) - \mu_i \\ \leq f_j(\theta) - \mu_j\}\end{aligned}$$
$$\begin{aligned}\mathcal{T} = \mathcal{T}(t) = \{J_i \,|\, \exists \theta \in [t-P,d], \forall j, f_i(\theta) \\ -\mu_i < f_j(\theta) - \mu_j\}\end{aligned}$$
$$\mathcal{M} = \mathcal{M}(t) = \mathcal{E} \cap \mathcal{T}$$

None of these sets is empty because they all contain the job J_i such that $f_i(d) - \mu_i$ is minimal. Let us define as $J_{i^\star}$ the job in $\mathcal{M}$ whose earliness cost α_i is minimal. We first show that the tardiness cost $\beta_{i^\star}$ of this job is also minimal in $\mathcal{M}$, that is $\beta_{i^\star} = \min\{\beta_i, J_i \in \mathcal{M}\}$. From the definition of $\mathcal{E}$, for any job $J_i \in \mathcal{M} - \{J_{i^\star}\}$, we have that $f_i(\theta) - \mu_i \leq f_{i^\star}(\theta) - \mu_{i^\star}$ for some $\theta \leq d$. As $\alpha_{i^\star} \leq \alpha_i$, we have $f_i(d) - \mu_i \leq f_{i^\star}(d) - \mu_{i^\star}$. If we had $\beta_i < \beta_{i^\star}$, we would have that $f_i(\theta) - \mu_i \leq f_{i^\star}(\theta) - \mu_{i^\star}$ for any $\theta \geq d$ and $J_{i^\star} \notin \mathcal{T}$, which is a contradiction. Therefore $\beta_{i^\star} \leq \beta_i$ and $\beta_{i^\star} = \min\{\beta_i, J_i \in \mathcal{M}\}$.

Then, we can define the set $\bar{\mathcal{E}} = \{J_i \in \mathcal{E} \,|\, \alpha_i < \alpha_i^\star\}$ and the time point $t^- = t_{i^\star}^-$. By definition of $J_{i^\star}$, all the jobs of $\bar{\mathcal{E}}$ are wholly early and scheduled in the interval $I(\bar{\mathcal{E}}) = [t-P, t^-]$. Symmetrically, let $\bar{\mathcal{T}}$ be the set $\{J_i \in \mathcal{T} \,|\, \beta_i < \beta_i^\star\}$, these jobs are scheduled in the interval $I(\bar{\mathcal{T}}) = [t^+, t]$ with $t^+ = t_{i^\star}^+$. We also observe that, since the tardy jobs are sequenced according to the tardiness costs, the tardy part of $J_{i^\star}$ is scheduled just before $\bar{\mathcal{T}}$, in the interval $[t^+ - p_{i^\star}^+, t^+]$. Therefore, for any $\epsilon \in [0, p_{i^\star}^+]$, we can build a feasible schedule for $\mathcal{P}(t-\epsilon)$ by left-shifting the jobs of $\bar{\mathcal{E}} \cup \bar{\mathcal{T}}$ by ϵ and by moving ϵ units of $J_{i^\star}$ from the interval $[t^+ - \epsilon, t^+)$ to $[t^- - \epsilon, t^-)$ (see Figure 1). A simple calculation shows that the cost of a job J_i in $\bar{\mathcal{E}}$ (resp. in $\bar{\mathcal{T}}$ increases by $\alpha_i p_i \epsilon$ (resp. decreased by $-\beta_i p_i \epsilon$). Then, the change of the cost of the schedule with respect to ϵ is

$$\sum_{J_i \in \bar{\mathcal{E}}} \alpha_i p_i \epsilon - \sum_{J_i \in \bar{\mathcal{T}}} \beta_i p_i \epsilon + \int_{t^- - \epsilon}^{t^-} f_{i^\star}(\theta)\mathrm{d}\theta - \int_{t^+ - \epsilon}^{t^+} f_{i^\star}(\theta)\mathrm{d}\theta \quad (6)$$

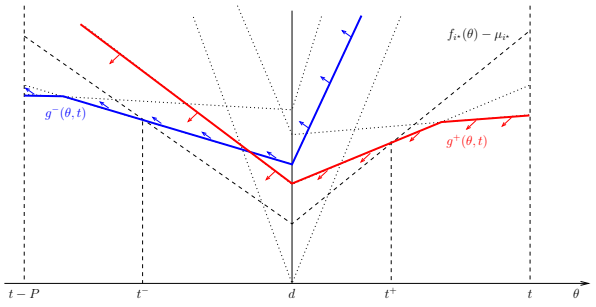

Fig. 2. The functions $\theta \mapsto g^-(\theta,t)$ and $\theta \mapsto g^+(\theta,t)$

We now show that, if ϵ is sufficiently small (less than some value δ to be determined), then the constructed schedule is optimal for $\mathcal{P}(t-\epsilon)$. To this end, we build the following dual feasible solution:

$$\mu_i(t-\epsilon) = \begin{cases} \mu_i(t) - (\alpha_{i^\star} - \alpha_i)\epsilon & \text{if } J_i \in \bar{\mathcal{E}}, \\ \mu_i(t) + (\beta_{i^\star} - \beta_i)\epsilon & \text{if } J_i \in \bar{\mathcal{T}}, \\ \mu_i(t) & \text{otherwise.} \end{cases}$$

The end of the proof is to evaluate the variation of the cost of the dual solution $q_{t-\epsilon}(\mu_i(t-\epsilon)) - q_t(\mu_i(t))$ in order to show that it matches the cost of the primal solution. We only give the main steps of the calculation. Let us consider the two functions depicted in Figure 2:

$$g^-(\theta,t) = \min_{J_i \in \bar{\mathcal{E}}}(f_i(\theta) - \mu_i(t))$$
$$g^+(\theta,t) = \min_{J_i \in \bar{\mathcal{T}}}(f_i(\theta) - \mu_i(t))$$

Simple calculations show that, for all $\theta \in I(\bar{\mathcal{E}})$, $g^-(\theta-\epsilon, t-\epsilon) = g^-(\theta,t) + \alpha_{i^\star}\epsilon$ and, for all $\theta \in I(\bar{\mathcal{T}})$, $g^+(\theta-\epsilon, t-\epsilon) = g^+(\theta,t) - \beta_{i^\star}\epsilon$. By construction of $\bar{\mathcal{E}}$ and $\bar{\mathcal{T}}$, we have that, for all $\theta \in I(\bar{\mathcal{E}})$, $g^-(\theta,t) < g^+(\theta,t)$. By continuity of g^+, there exists $\delta' > 0$ such that, for any $\epsilon \in [0,\delta']$, we have $g^+(\theta, t-\epsilon) > \min_{J_i \notin \bar{\mathcal{T}}}(f_i(\theta-\epsilon) - \mu_i(t-\epsilon))$. Then,

for $\epsilon \leq \delta'$, we have (detailed calculations are omitted):

$$\min_{1\leq i\leq n}(f_i(\theta-\epsilon)-\mu_i(t-\epsilon)) = \begin{cases} g^-(\theta-\epsilon,t-\epsilon)=g^-(\theta,t)+\alpha_{i^\star}\epsilon & \text{if } \theta\in I(\bar{\mathcal{E}}) \\ g^+(\theta-\epsilon,t-\epsilon)=g^+(\theta,t)-\beta_{i^\star}\epsilon & \text{if } \theta\in I(\bar{\mathcal{T}}) \\ \min_{1\leq i\leq n}(f_i(\theta-\epsilon)-\mu_i(t)) & \text{otherwise.} \end{cases}$$

Using this result, we can compute

$$\begin{aligned}\int_{t-P-\epsilon}^{t-\epsilon} &\min_{J_i\in\bar{\mathcal{T}}}(f_i(\theta)-\mu_i(t-\epsilon))\mathrm{d}\theta \\ &= \int_{t-P}^{t}\min_{J_i\in\bar{\mathcal{T}}}(f_i(\theta-\epsilon)-\mu_i(t-\epsilon))\mathrm{d}\theta \\ &= \int_{t-P}^{t}\min_{J_i\in\bar{\mathcal{T}}}(f_i(\theta)-\mu_i(t))\mathrm{d}\theta+\alpha_{i^\star}\sum_{J_i\in\bar{\mathcal{E}}}p_i\epsilon \\ &\quad -\beta_{i^\star}\sum_{J_i\in\bar{\mathcal{T}}}p_i\epsilon+\int_{t^--\epsilon}^{t^-}f_{i^\star}-\int_{t^+-\epsilon}^{t^+}f_{i^\star}\end{aligned}$$

So, we finally have

$$\begin{aligned}q_{t-\epsilon}(\mu_i(t-\epsilon))-q_t(\mu_i(t)) &= \sum_{J_i\in\bar{\mathcal{E}}}\alpha_i p_i\epsilon \\ &\quad -\sum_{J_i\in\bar{\mathcal{T}}}\beta_i p_i\epsilon+\int_{t^--\epsilon}^{t^-}f_{i^\star}-\int_{t^+-\epsilon}^{t^+}f_{i^\star} \quad (7)\end{aligned}$$

We have then proved that the variation of the cost of the dual solution is equal to the variation of the cost of the primal solution given by (6) so that the proposed primal solution is optimal. Therefore, we have shown that if we choose $J_{i^\star}$ and $\delta=\min(p^+_{i^\star},\delta')$, the theorem is valid.

The theorem gives the main idea of the algorithm to compute all the values $\mathrm{OPT}(t)$ when t varies. If we first consider a "continuous" version of the algorithm, we have to determine at any time t the transfered job $J_{i^\star}$. The proof of Theorem 1 shows how to select this job but, in order to have a simpler algorithm, we give a second characterization of the transfered job. We consider the change of the cost of the schedule, when the quantity $\epsilon\in(0,p_i^+]$ of a job J_i is transfered.

$$\sum_{\alpha_j<\alpha_i}\alpha_j p_j^-\epsilon-\sum_{\beta_j<\beta_i}\beta_j p_j^+\epsilon+\int_{t_i^--\epsilon}^{t_i^-}f_i(\theta)\mathrm{d}\theta-\int_{t_i^+-\epsilon}^{t_i^+}f_i(\theta)\mathrm{d}\theta.$$

Since ϵ can be arbitrarily small, we define the *marginal transfer cost* m_i *of* J_i that correspond to the limit of the transfer cost when ϵ tends to 0^+:

$$\begin{aligned}m_i &= \sum_{\alpha_j<\alpha_i}\alpha_j p_j^- - \sum_{\beta_j<\beta_i}\beta_j p_j^+ + \alpha_i\left(\sum_{\alpha_j\geq\alpha_i}p_j^-\right) \\ &\quad -\beta_i\left(\sum_{\beta_j\geq\beta_i}p_j^+\right) \\ &= \sum_j \min(\alpha_j,\alpha_i)p_j^- - \min(\beta_j,\beta_i)p_j^+ \quad (8)\end{aligned}$$

In order that the schedule obtained after the transfer of a very small quantity ϵ of J_i is optimal, the transfered job must be the job with the minimal transfer cost m_i. If there are several jobs that minimize m_i, the proof of Theorem 1 shows that we must select the job with the smallest value α_i.

Corollary 2 *The function $t\mapsto\mathrm{OPT}(t)$ is convex.*

PROOF. From the definition of the marginal costs m_i, we have that the derivative $\mathrm{OPT}'(t)=-\min_{1\leq i\leq n}m_i(t)$. For any t,t' such that $\max(P,d)\leq t\leq t'\leq d+P$, we have, for any j, $p_j^+(t)\leq p_j^+(t')$ and $p_j^-(t)\geq p_j^-(t')$. Therefore, for any i, $m_i(t)\geq m_i(t')$ and therefore $\mathrm{OPT}'(t)\leq\mathrm{OPT}'(t')$, which proves the convexity of OPT.

Once the transfered job $J_{i^\star}$ is selected, we must determine the quantity to be transfered. Clearly, the transfer of $J_{i^\star}$ must be stopped when one of the four following events is met:

(1) $J_{i^\star}$ has been wholly transfered (that is $p^+_{i^\star}$ becomes null),
(2) when there is no room left in $[0,d]$ to realize the transfer,
(3) when the minimum of $\mathrm{OPT}(t)$ has been reached,
(4) when another job becomes more critical.

We now study the fourth type of events and, to this end, we consider the variation of the transfer costs while $J_{i^\star}$ is transfered. Since only $p^-_{i^\star}$ and $p^+_{i^\star}$ in (8) are modified during the transfer, we have for any J_i with $p_i^+>0$:

$$\begin{aligned}m_i(t-\epsilon) &= m_i(t)+s_i(t)\epsilon \quad \text{with} \\ s_i(t) &= \min(\alpha_i,\alpha_{i^\star})+\min(\beta_i,\beta_{i^\star}) \quad (9)\end{aligned}$$

Therefore, the variation of the transfer cost of each job is linear. Clearly, if $s_i>s_{i^\star}$, we will have $m_i(t-\epsilon)\geq m_{i^\star}(t-\epsilon)$ for any $\epsilon>0$ and, conversely, by considering

the cases $s_i < s_{i^\star}$, we find that $m_{i^\star}(t-\epsilon)$ is ensured to be less than $m_i(t-\epsilon)$ as long as

$$\epsilon \leq \min\left\{ \frac{m_i - m_{i^\star}}{s_{i^\star} - s_i} \,,\, p_i^+ > 0 \text{ and } s_i < s_{i^\star} \right\}.$$

We now consider the case where the optimum of the problem is reached after that the quantity ϵ of $J_{i^\star}$ was transfered (third type of events). While $J_{i^\star}$ is being transfered, since the derivative of the optimal cost is $\mathrm{OPT}'(t) = -m_{i^\star}(t)$, the event cannot happen unless $m_{i^\star}(t-\epsilon) = 0$, that is $\epsilon = -\frac{m_{i^\star}}{s_{i^\star}} = \frac{-m_{i^\star}}{\alpha_{i^\star}+\beta_{i^\star}}$ (note that $m_{i^\star} \leq 0$ and $\alpha_{i^\star} + \beta_{i^\star} > 0$).

The first two events are obviously detected in $O(1)$ time and we can now write Algorithm 2 that computes the optimal schedule for the problem. The main loop corresponds to the events that are iteratively met while the jobs are transfered.

Algorithm 2 Solve the common due date problem

$t \leftarrow d + P$
for each i: let $p_i^+ = p_i$
for each i: initialize $m_i = -\sum_{j=1}^{n} \min(\beta_i, \beta_j) p_j$
repeat
  let $J_{i^\star}$ be the job that minimizes (m_i, α_i, β_i) in the lexicographical order
  for each i: compute $s_i = \min(\alpha_i, \alpha_{i^\star}) - \min(\beta_i, \beta_{i^\star})$
  let $\delta = \min\left(p_{i^\star}^+, t-P, \min\left\{ \frac{m_i - m_{i^\star}}{s_{i^\star} - s_i} \,,\, p_i^+ > 0 \text{ and } s_i < s_{i^\star} \right\}, \frac{-m_{i^\star}}{s_{i^\star}}\right)$
  $t = t - \delta$
  decrease $p_{i^\star}^+$ by δ
  for each i: update $m_i = m_i + s_i\delta$
until $t = \max(P, d)$ or $m_{i^\star} = 0$
build the schedule according to the p_i^+ and compute its cost

The main difficulty to analyze the complexity of this algorithm is to bound the number of events that may occur. To this end, we first prove the following lemma.

Lemma 3 *When the transfered job $J_{i^\star}$ is changed but $p_{i^\star}^+ > 0$, the new transfered job J_j is wholly late ($p_j^+ = p_j$).*

PROOF. According to the proof of Theorem 1, for any job J_i with $p_i^- > 0$ and $p_i^+ > 0$, we have $\alpha_i \geq \alpha_{i^\star}$ and $\beta_i \geq \beta_{i^\star}$. Therefore, by (9), $s_i \geq s_{i^\star}$, which means that the transfer cost $m_i(t-\epsilon)$ of J_i cannot become less than $m_{i^\star}(t-\epsilon)$ while $J_{i^\star}$ has not been wholly transfered.

Theorem 4 *The common due date scheduling problem can be solved in $O(n^2)$ time.*

PROOF. The first event can happen at most n times, the second and the third at most once and the fourth at most n times according to the previous lemma. Consequently, the main loop of the algorithm is run $O(n)$ times. Clearly the instructions inside the loop require $O(n)$ time. Furthermore, the initialization of the marginal transfer costs is done in $O(n^2)$ time. Then, the time complexity of the proposed algorithm is $O(n^2)$.

In the optimal schedule built by the algorithm, each job is interrupted at most once but the job in process at time d is not interrupted, so there are at most $n-1$ interruptions in the schedule.

In the special case where $\alpha_i = \beta_i$ for all $i \in \{1, \ldots, n\}$, the problem can be solved in $O(n \log n)$. First, if the common due date d is unrestricted, the problem is clearly symmetric and there is an optimal schedule such that $p_i^+ = p_i^- = p_i/2$ for each job J_i. The corresponding schedule can be scheduled in $O(n \log n)$ and it is optimal whenever $d \geq P/2$. We observe that this schedule has exactly $n-1$ interruptions. If $d < P/2$, we first order the jobs in the decreasing order of the α_i's and we then compute the vector p^- (and thereafter p^+) by Algorithm 3.

Algorithm 3 Solve the common due date problem with symmetric costs

$i = 1$
$t^- = P$
repeat
  $p_i^- = \min(p_i/2, t^-)$
  decrease t^- by p_i^- and increase i
until $t^- = 0$ or $i = n$
while $i \leq n$ **do**
  $p_i^- = 0$
  increase i
end while

A last special case is when the jobs can be renumbered such that $\alpha_1 \leq \cdots \leq \alpha_n$ and $\beta_1 \geq \cdots \geq \beta_n$. Then, the transfer of a job is never interrupted and the schedule has no preemption (the jobs are sequenced in the order of their indices). Since the job are simply transfered in the same order, the choice of the transfered job is easy and the fourth event cannot happened. Therefore, Algorithm 2 can be simplified to run in $O(n)$ time (once the jobs are sorted).

4. A two-machine problem

When preemption is not allowed, we often have a strong relationship between one-machine scheduling around a common due date and two-machine scheduling: the schedule of the first (resp. second) machine in the two-machine problem corresponds to the late (resp. early) jobs in the common due date problem. However, this relationship disappears in preemptive scheduling because we have the constraint that a job cannot be scheduled on two different machines at a single time. This constraint makes the preemptive two-machine deeply different from the problem addressed in Section 3.2..

This section is then devoted to the two-machine problem. The n jobs $J_1, \dots, J_n$ have to be processed by $m = 2$ identical parallel machines. They are all available at time 0. As in Section 3.1., we consider that $f_i(t) = w_i t$ with $w_i > 0$. We also assume that the jobs are non-increasingly sorted according to their slopes ($w_1 \geq \cdots \geq w_n > 0$).

As in the previous section, the algorithm is presented in the case where the slopes are assumed to be all different ($w_1 > \cdots > w_n$) in order to avoid some technicalities in the proofs. If $w_i = w_{i+1}$ for some job J_i, we can change the weight $w_i = w_i + \epsilon$ for a very small ϵ.

4.1. *Dominance properties*

Any feasible schedule can be described by the two functions $\mathcal{J}_1$ and $\mathcal{J}_2 : \mathbb{R}_+ \to \{1, \dots, n, \infty\}$. Abusing notation, $\mathcal{J}_j(t)$ will both denote the job running at t on machine j and its index. $\mathcal{J}_j(t) = \infty$ means that no job is processed. As the objective function is regular and there is no release dates, there clearly exists an optimal solution with no idle time inserted, which means that if $\mathcal{J}_j(t) = \infty$ for some t, then $\mathcal{J}_j(t') = \infty$ for all $t' \geq t$.

Lemma 5 *There is an optimal schedule such that if $\mathcal{J}_1(t) = \mathcal{J}_1(t')$ and $\mathcal{J}_2(t) = \mathcal{J}_2(t')$ for some $t < t'$ then $\mathcal{J}_1(\theta) = \mathcal{J}_1(t)$ and $\mathcal{J}_2(\theta) = \mathcal{J}_2(t)$ for all $\theta \in [t, t']$.*

PROOF. The proof is based on the simple interchange argument illustrated by Figure 3. Job J_i and J_j are respectively scheduled on machine 1 and 2 during the intervals (t_1, t'_1) and (t_2, t'_2). Let B_1 and B_2 be the parts of jobs processed in between the two intervals on machine 1 and 2 respectively. Let p_k^B be the amount (in processing time) of job J_k scheduled between t'_1 and t'_2 in B_1 or in B_2. Let $W^B = \sum_{k=1}^{n} w_k p_k^B$. Simple calculations show that if we swap the left part of J_i with B_1 and the left part of J_j with B_2 the variation of the cost is equal to

$$-\left(W^B - (w_i + w_j)(t_2 - t'_1)\right)(t'_1 - t_1)$$

Similarly, if we swap the right part of J_i with B_1 and the right part of J_j with B_2 the variation of the cost is equal to

$$\left(W^B - (w_i + w_j)(t_2 - t'_1)\right)(t'_2 - t_2)$$

Therefore either one move improves the schedule or both moves let the cost unchanged, which means that at least one of the two interchanges does not increase the cost. By iterating such interchanges, we prove that an optimal schedule satisfies the conditions of the lemma.

We observe that the schedules that satisfy the conditions of the lemma have at most $n(n+1)$ job interruptions since there are at most $n(n+1)$ different job assignments for the two machines. The following lemma reinforces the dominance properties.

Lemma 6 *There is an optimal schedule with no idle time such that the two following conditions hold:*
(1) for any $t \geq 0$, $\mathcal{J}_1(t) < \mathcal{J}_2(t)$
(2) for any $0 \leq t < t'$ and $j \in \{1, 2\}$, $\mathcal{J}_j(t) \leq \mathcal{J}_j(t')$

PROOF. From any optimal schedule, another optimal schedule satisfying $\mathcal{J}_1(t) < \mathcal{J}_2(t)$ can be built by swapping the parts of jobs between the machines. Therefore, we assume we have an optimal schedule that satisfies the first property. Let $\mathcal{S}$ be the set of optimal schedule with no idle time that satisfy the first property and the conditions of Lemma 5. Since these schedule have at most $n(n+1)$ interruptions, the set $\mathcal{S}$ is compact. For each schedule in $\mathcal{S}$, let us consider the time t of the *earliest violation* of the second condition, that is the earliest t such that there exists $t' > t$ such that $\mathcal{J}_1(t) > \mathcal{J}_1(t')$ or $\mathcal{J}_2(t) > \mathcal{J}_2(t')$. If there is no schedule that satisfy the second condition, all the schedules have an earliest violation and we consider the schedule with the latest earliest violation t (such a schedule exists since $\mathcal{S}$ is compact).

Let ϵ be such that there is no preemption in the two intervals $[t, t+\epsilon)$ and $[t', t'+\epsilon)$. For example, $\mathcal{J}_1(t)$ denotes the *only* job scheduled on machine 1 in time interval $[t, t+\epsilon)$ and, since there is no ambiguity, $\mathcal{J}_1(t)$ is also used to denote the part of job $\mathcal{J}_1(t)$ scheduled in the interval $[t, t+\epsilon)$.

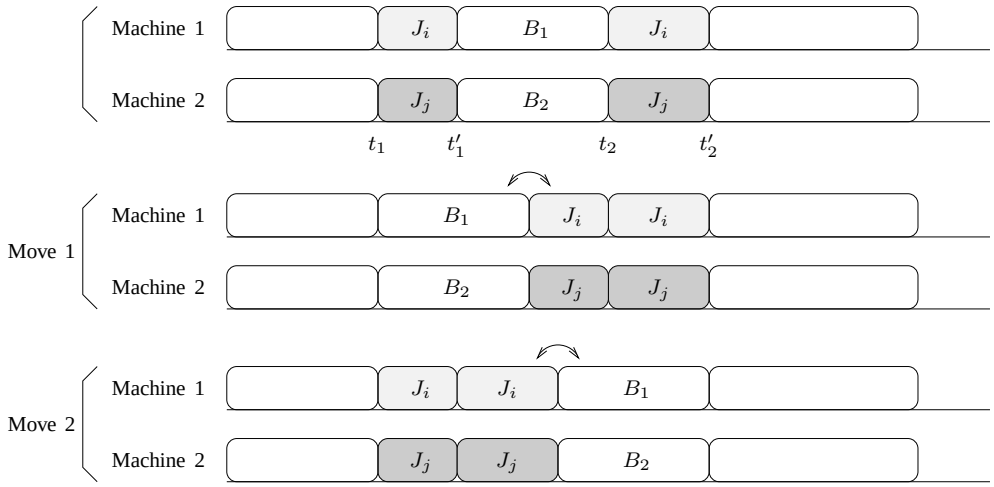

Fig. 3. Proof of Lemma 5

- If there is a violation on both machines, that is $\mathcal{J}_1(t) > \mathcal{J}_1(t')$ and $\mathcal{J}_2(t) > \mathcal{J}_2(t')$, we can swap $\mathcal{J}_1(t)$ with $\mathcal{J}_1(t')$ and, simultaneously, $\mathcal{J}_2(t)$ with $\mathcal{J}_2(t')$. The swap operation does not increase the cost, does not violate the first condition but removes the violation of the second condition at time t, which contradicts the maximality of t.
- If $\mathcal{J}_1(t) > \mathcal{J}_1(t')$ and $\mathcal{J}_2(t) \leq \mathcal{J}_2(t')$, we have that $\mathcal{J}_2(t') \geq \mathcal{J}_2(t) > \mathcal{J}_1(t) > \mathcal{J}_1(t')$ so that $\mathcal{J}_1(t)$ and $\mathcal{J}_1(t')$ can be swapped without violating the first constraint. Once again, the violation of the second condition at time t disappears.
- If $\mathcal{J}_1(t) \leq \mathcal{J}_1(t')$ and $\mathcal{J}_2(t) > \mathcal{J}_2(t')$, we symmetrically have that $\mathcal{J}_1(t') \geq \mathcal{J}_1(t) > \mathcal{J}_2(t) > \mathcal{J}_2(t')$ and we have the same conclusion when swapping $\mathcal{J}_2(t)$ and $\mathcal{J}_2(t')$.

As the maximality of t is contradicted in every case, there is an optimal schedule that satisfy the two conditions.

Even if our approach for solving is based on dynamic programming, we first introduce a quadratic programming formulation of the problem which will be useful to clearly define the subproblems of the dynamic program.

4.2. *Quadratic programming formulation*

As a corollary of Lemma 6, job J_1 is scheduled without preemption on the first machine between 0 and p_1 — hence its cost is ${w_1}{p_1}^2/2$. Similarly, job J_n is also scheduled in an interval of length p_n and it can be completely scheduled on the second machine (see Figure 4).

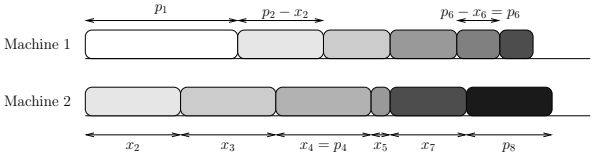

Fig. 4. Variables of the quadratic program

Note that, in order to satisfy the conditions of Lemma 6, the part of J_n scheduled after the completion of the first machine should be moved from machine 2 to machine 1.

A dominant schedule with n jobs is then described by $n-2$ variables $x_2, \ldots, x_{n-1}$ where x_i is the part of job J_i schedule on the second machine. For convenience, we introduce the variables $X_i = \sum_{j=1}^{i} x_j$ (with $X_1 = 0$). According to the dominance rules, J_i $(1 < i < n)$ starts on the second machine at X_{i-1} and is interrupted at X_i. Then, with the notation $P_i = \sum_{j=1}^{i} p_j$, the processing is resumed on the first machine at $P_{i-1} - X_{i-1}$ until it completes at $P_i - X_i$. Finally, J_n is processed between X_{n-1} and $X_{n-1} + p_n$. Our problem can now be formulated as a quadratic program QP.

$$\begin{aligned}\min \big({w_1}{p_1}^2/2 &+ \sum_{i=2}^{n} w_i(X_{i-1} + x_i/2)x_i \\ &+ \sum_{i=2}^{n} w_i(P_{i-1} - X_{i-1} + (p_i - x_i)/2)(p_i - x_i) \\ &+ w_n(X_{n-1} + p_n/2)p_n \big) \quad (10)\end{aligned}$$

$$\text{s.t.} \quad \forall 1 < i < n \qquad 0 \leq x_i \leq p_i \tag{11}$$

$$\forall 1 \leq i < n \qquad X_i = \sum_{j=1}^{i} x_j \tag{12}$$

$$\forall 1 < i < n \qquad 2X_{i-1} + x_i \leq P_{i-1} \tag{13}$$

The objective function (10) can be expressed in function of the x_i's using (12) so that it becomes a quadratic polynomial in $x_2, \ldots, x_{n-1}$ where all the quadratic terms have a nonnegative coefficient. Equations (13) mean that J_i can start on the first machine only after it is stopped on the second machine.

4.3. *Dynamic programming recursion*

In this section, we propose a dynamic programming approach to solve the two-machine problem. The main idea is similar to other classic dynamic programming algorithms for two-machine problems (for example $P2||C_{\max}$ [5]). The jobs are added to the partial schedule in the order of their slopes and for each partial schedule, the dynamic program records all the optimal solutions for all the possible completion time of the second machine. More formally, let $f_k(t)$ be the minimal cost for scheduling the first k jobs such that the second machine completes at t (which means that the first machine completes at $P_k - t$). While scheduling algorithms usually consider all the possible integer values of t, we here consider all the real values of t. It means that the functions f_k will not be recorded as an array of values but we will use a more compact data structure using the property that the functions are piecewise quadratic.

By an immediate adaptation of the proof of Lemma 6, it can be shown that the dominance properties remain valid when this new constraint on the completion time of the second machine is added. Therefore we can modify QP in order that $f_k(t)$ is given by the optimum of the quadratic program $\mathrm{QP}_k(t)$:

$$\min \big({w_1 {p_1}^2}/{2} + \sum_{i=2}^{k} w_i (X_{i-1} + x_i/2) x_i \\ \sum_{i=2}^{k} w_i (P_{i-1} - X_{i-1} + (p_i - x_i)/2)(p_i - x_i) \big) \tag{14}$$

$$\text{s.t.} \quad 0 \leq x_i \leq p_i \quad \forall 1 < i \leq k \tag{15}$$

$$X_i = \textstyle\sum_{j=1}^{i} x_j \qquad \forall 1 \leq i \leq k \tag{16}$$

$$2X_{i-1} + x_i \leq P_{i-1} \qquad \forall 1 < i \leq k \tag{17}$$

$$X_k \leq t \tag{18}$$

The objective function takes into account that the job J_k can now be preempted (note also the presence of the variable x_k). Equations (15), (16) and (17) respectively correspond to (11), (12) and (13). Finally, equation (18) indicates that the second machine must complete before t. Clearly, all the functions $f_k(t)$ become constant when t becomes large, let T_k be the smallest time point such that f_k is constant on the interval $[T_k, \infty)$. For any k, we observe that $0 \leq T_k \leq P_k/2$.

For the purpose of illustration, let us first consider $f_1(t)$. Clearly, J_1 is scheduled without preemption on M_1 so that $f_1(t) = \int_0^{p_1} w_1 \theta \mathrm{d}\theta = \frac{1}{2} w_1 {p_1}^2$ for any t, which means that $T_1 = 0$. When a second job is added, the job can be scheduled without interruption between 0 and $\min(p_1, t)$ on M_2. If the job J_2 is not completed at this time, J_2 is interrupted and restarts on M_1 at p_1. Therefore, simple calculations give

$$f_2(t) = \begin{cases} \frac{1}{2}(w_1 {p_1}^2 + w_2 {p_2}^2) & \text{if } t \geq \min(p_1, p_2) \\ \frac{1}{2}\big(w_1 {p_1}^2 & \\ +w_2(2t^2 - 2t(p_1 + p_2) & \\ +2p_1 p_2 + {p_2}^2)\big) & \text{otherwise} \end{cases}$$

and the derivative is

$$f_2'(t) = \begin{cases} 0 & \text{if } t > \min(p_1, p_2) \\ w_2(2t - P_2) & \text{if } t < \min(p_1, p_2) \end{cases}$$

Clearly, we have $T_2 = \min(p_1, p_2)$. If $p_1 \neq p_2$, $f'(T_2)$ does not exist but the left and right derivatives exist. As $T_2 \leq P_2$, we have that $f_2'(t) \leq 0$ and f_2' is piecewise linear and nondecreasing. Therefore, the function f_2 is nonincreasing, convex and piecewise quadratic. Furthermore, $f_2''(t) = 2w_2$ for any $t \in [0, T_2)$.

We now show by induction that, for any $k \geq 2$, we have the three following properties:

(1) the function f_k is nonincreasing, convex and piecewise quadratic,
(2) the derivative f_k' is nondecreasing and piecewise linear and continuous on $[0, T_k)$ and $f_k'(0) \leq -w_k P_k$,
(3) the second derivative f_k'' is piecewise constant and for any $t \in [0, T_k)$, $f_k''(t) \in \{2w_2, \ldots, 2w_k\}$ (if it is defined) and $f_k''(t) = 0$ for $t > T_k$.

Clearly, the three properties are true for $k = 2$ (for $k = 1$, we have $f'1(0) = 0 \not\leq -w_1 P_1$ but the other

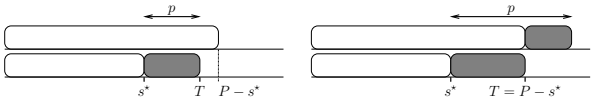

Fig. 5. The two cases when computing T

properties are satisfied). We then assume that, for some $k > 2$, they are true for f_{k-1}, f'_{k-1} and f''_{k-1} and we are going to build f_k and show that it satisfies the above properties. In order to have simpler notations, let us define $f = f_{k-1}$, $P = P_{k-1}$, $J = J_k$, $w = w_k$, $p = p_k$, $F = f_k$ and $T = T_k$. Let s denote the start time of J on the second machine.

We first compute the time T when F becomes constant. We then solve the problem $\mathrm{QP}_k(\infty)$, which correspond to the removal of equation (18). It was noted in the previous subsection that, in an optimal solution of this problem, the last job, that is J, is not interrupted. Therefore, the cost of the schedule is $f(s)+\int_s^{s+p}\theta \mathrm{d}\theta = f(s)+wp(s+p/2)$ where s is the start time of J. Since f has both left and right derivative — respectively denoted by f'_- and f'_+ — and f' is nondecreasing, the start time s must satisfy $f'_-(s) \leq -wp \leq f'_+(s)$ in the optimal schedule. As $-wp < 0$, we have $s < T_{k-1}$ and there is a unique solution, denoted by $s^\star$ — we conventionally define $f'_-(0) = -\infty$ to ensure the existence of $s^\star$. Figure 5 represents the two cases that can happen: either job J completes before machine 1 completion (so that $T = s^\star + p$) or it completes later (so that $T = P - s^\star$).

In an optimal solution of $\mathrm{QP}_k(t)$ with $t < T$, J starts at s on the second machine and is interrupted at t then it is resumed at $P - s$ on the first machine and completes at $P + p - t$ (possibly, $s = t$ or $P - s = P + p - t$). Therefore, the cost of the schedule is

$$\begin{aligned} F(t) = F(s,t) &= f(s) + w\int_s^t \theta \mathrm{d}\theta + w\int_{P-s}^{P+p-t} \theta \mathrm{d}\theta \\ &= f(s) + \frac{w}{2} \\ &\quad \left(t^2 - s^2 + (P+p-t)^2 - (P-s)^2\right) \quad (19) \end{aligned}$$

In fact, the variable s is subject to feasibility constraints so that $F(t)$ can also be expressed as the following mathematical program, which is in fact a unidimensional parameterized problem (the variable is s while t is a fixed parameter in $[0, T]$).

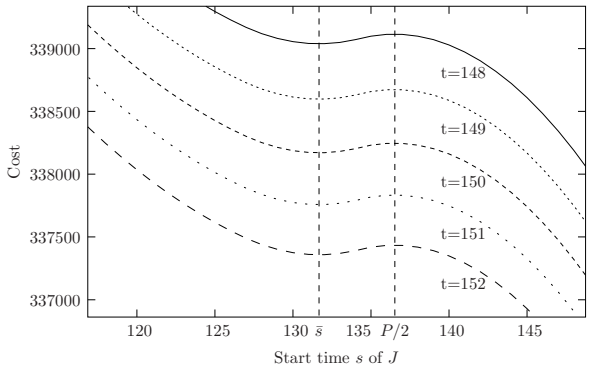

Fig. 6. Functions φ_t for different values of t

$$\begin{aligned} \min \quad & F(s,t) && (20) \\ \text{s.t.} \quad & t - p \leq s \leq t && (21) \\ & s \geq 0 && (22) \\ & t \leq P - s && (23) \end{aligned}$$

Equation (21) indicates that the part of J scheduled on the second machine is not greater than p. Equation (22) forces J to start after 0 while equation (23) prevents J from starting on machine 1 before it is completed on machine 2. The problem is then to minimize the function $\varphi_t : s \mapsto \varphi_t(s) = F(s,t)$ on the interval $I_t = [\max(0, t-p), \min(t, P-t)]$. We note that this interval is not empty because $t \geq \max(0, t-p)$ and $P - t = P - 2t + t \geq P - 2T + t \geq P - 2\frac{P+p}{2} + t = t - p$.

Disregarding constraints (21)-(23), let us consider the partial derivative

$$\begin{aligned} \varphi'_t(s) = \frac{\partial F(s,t)}{\partial s} &= f'(s) + \frac{w}{2}\left(-2s + 2(P-s)\right) \\ &= f'(s) + w(P - 2s) \quad (24) \end{aligned}$$

Note that $f'(s)$ and thereafter $\varphi'_t(s)$ may not exist at some points but the left and right derivatives exist. For simplicity and as long as it is unambiguous, we only write one derivative instead of defining both left and right derivatives.

Clearly, for any t, the derivative is null for $s = P/2$ which corresponds to a local maximum of φ'_t (see Figure 6). Since f' is nondecreasing and $f'(0) \leq -w_{k-1}P \leq -wP$ and $f'(T) = 0 \geq w(2T - P)$, there is at least one value $s \in [0, T]$ such that $f'_-(s) \leq w(2s - P) \leq f'_+(s)$. Moreover, as $f''(t) > 2w$ for any $t \in [0, T]$, this solution is unique. Let us denote it by $\bar{s}$. Clearly, $\bar{s}$ does not

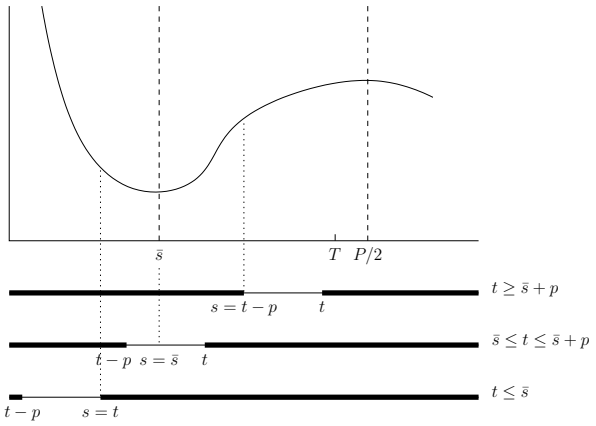

Fig. 7. Start time s according to t when $T < P - s^\star$

depend on t. In some degenerate cases, we can have $\bar{s} = T = P/2$ but when $T < P/2$, we have $\varphi'(t) > 0$ for all $t \in (T, P/2)$, which means that $\bar{s}$ is the unique global minimum of $\varphi'(t)$ in the interval $[0, P/2]$ (see Figure 6).

In order to study the minimization of φ_t on I_t, let us first calculate

$$\begin{aligned}\varphi_t'(s^\star) &= f'(s^\star) + wP - 2ws^\star \\ &= -wp + wP - 2ws^\star \\ &= 2w\left(\frac{P-p}{2} - s^\star\right)\end{aligned} \tag{25}$$

Let us now compute $F(t)$ in each of the two cases defined above (see Figure 5). Let us first consider that $T < P - s^\star$. According to (25), we have that $\varphi_t'(s^\star) > 0$ because $T = s^\star + p < P - s^\star$ gives $\frac{P-p}{2} - s^\star > 0$. Therefore, we have $\bar{s} < s^\star$ and $I_T \subset [\bar{s}, P/2]$. As $t < P - t$, the interval I_t is equal to $[\max(0, t-p), t]$. Then, as illustrated by Figure 7, the start time s of J is equal to

$$s = \begin{cases} t-p & \text{if } \bar{s}+p \leq t \leq T \\ \bar{s} & \text{if } \bar{s} \leq t \leq \bar{s}+p \\ t & \text{if } 0 \leq t \leq \bar{s} \end{cases} \tag{26}$$

The left column of Figure 8 illustrates how the scheduling of J varies when t decreases. From the value of s, the cost of the schedule is immediately derived.

$$F(t) = \begin{cases} f(t-p) + wp(t-p/2) \text{ if } \bar{s}+p \leq t \leq T \\ f(\bar{s}) + \frac{w}{2}\big(t^2 - \bar{s}^2 + (P+p-t)^2 \\ \qquad -(P-\bar{s})^2\big) \qquad \text{if } \bar{s} \leq t \leq \bar{s}+p \\ f(t) + wp(P-t+p/2) \text{ if } 0 \leq t \leq \bar{s} \end{cases} \tag{27}$$

and the derivative of this cost function is

$$F'(t) = \begin{cases} f'(t-p) + wp & \text{if } \bar{s}+p \leq t \leq T \\ 2w\left(t - \frac{P+p}{2}\right) & \text{if } \bar{s} \leq t \leq \bar{s}+p \\ f'(t) - wp & \text{if } 0 \leq t \leq \bar{s} \end{cases} \tag{28}$$

Let us now consider the second case $T = P - s^\star \leq s^\star + p$. According to (25), we now have that $\varphi_t'(s^\star) \leq 0$ and then $s^\star \leq \bar{s}$ and $P - \bar{s} \leq P - s^\star = T$.

$$s = \begin{cases} P-t & \text{if } P-\bar{s} \leq t \leq T \\ \bar{s} & \text{if } \bar{s} \leq t \leq P-\bar{s} \\ t & \text{if } 0 \leq t \leq \bar{s} \end{cases} \tag{29}$$

The right column of Figure 8 shows the corresponding schedules and the cost function and its derivative are equal to

$$F(t) = \begin{cases} f(P-t) + wp(P-t+p/2) \\ \qquad \text{if } P-\bar{s} \leq t \leq T \\ f(\bar{s}) + \frac{w}{2}\big(t^2 - \bar{s}^2 + (P+p-t)^2 \\ \quad -(P-\bar{s})^2\big) \quad \text{if } \bar{s} \leq t \leq P-\bar{s} \\ f(t) + wp(P-t+p/2) \quad \text{if } 0 \leq t \leq \bar{s} \end{cases} \tag{30}$$

and

$$F'(t) = \begin{cases} -f'(P-t) - wp & \text{if } P-\bar{s} \leq t \leq T \\ 2w\left(t - \frac{P+p}{2}\right) & \text{if } \bar{s} \leq t \leq P-\bar{s} \\ f'(t) - wp & \text{if } 0 \leq t \leq \bar{s} \end{cases} \tag{31}$$

For the functions F built in both cases, we clearly have

(1) the function F is nonincreasing, convex and piecewise quadratic,
(2) the derivative F' is nondecreasing, piecewise linear and continuous over $(0, T)$, and $F'(0) \leq -w(P+p)$,
(3) the second derivative F'' is piecewise constant and for any $t \in [0, T)$, $F''(t) \in \{2w_2, \ldots, 2w_k\}$ and $F''(t) = 0$ for $t > T$.

Therefore, we have proved by induction that the property is true for any function f_k.

4.4. *Algorithm*

In order to describe the algorithm that implements the dynamic programming scheme, the main work is to provide a data structure that encodes the functions f_k. Classically, piecewise linear functions are represented as a

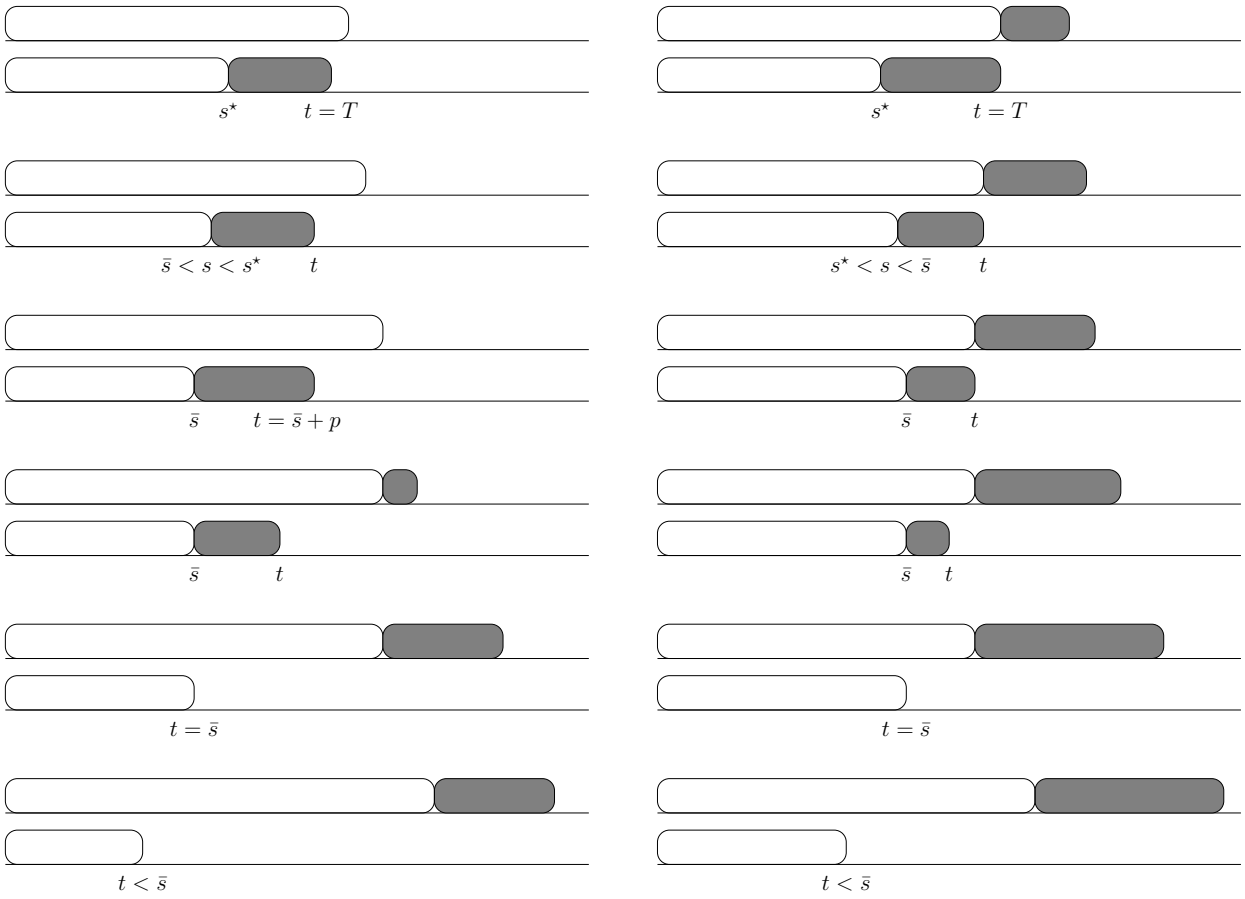

Fig. 8. Optimal scheduling of J when t varies

sorted list of its segments [8,17] and piecewise quadratic functions can similarly be represented by a sorted list: each quadratic piece of the function is encoded by a cell of the list that contains the left and right endpoints of the segment (l, r) and the three values (α, β, γ) such that the function is equal to $x \mapsto \alpha x^2 + \beta x + \gamma$ on the interval (l, r).

However, in the present algorithm, this encoding of the function could lead to a non-polynomial algorithm. To illustrate the point, let us use the notation of the previous subsection and remind that $\bar{s}$ and $s^\star$ are such that

$$f'_-(\bar{s}) \leq w(2\bar{s} - P) \leq f'_+(\bar{s}) \quad \text{and}$$
$$f'_-(s^\star) \leq -wp \leq f'_+(s^\star).$$

Consequently, assuming we are free to choose the processing time p and the weight w of J, we can first select a sufficiently small value for w such that $\bar{s}$ is later than the latest breakpoint of f. Then, by selecting p larger than $-f'_+(0)/w$, we have $s^\star = 0$. Since $-f'_+(0)/w \geq P$, the processing time p is larger than P. Therefore, F is built according to (30). As the abscissas of the breakpoints of f are all in the interval $[s^\star, \bar{s}]$, a breakpoint of f with abscissa b will cause two breakpoints with abscissa b and $P - b$ in the new functions F. Accordingly, the number of breakpoints of F is at least twice the number of breakpoints of f. By iterating this construction procedure, we can then build instances in which the functions f_k have an exponential number of quadratic pieces. To avoid the problem, we adopt a non-explicit but more compact encoding of our cost functions.

Each function f_k is encoded by the following values, which can be stored in simple arrays:

- the value T_k. If $T_k \leq P_k/2$, f_k was build according to equation (27) (first case). Otherwise, it was build according to equation (30) (second case).

- an interval (l_k, r_k) with $l_k = \bar{s}_k$ and $r_k = \bar{s}_k + p_k$ in the first case or $r_k = P_k - \bar{s}_k$ in the second case ($\bar{s}_k$ corresponds to $\bar{s}$ when $F = f_k$). If $k = 1$, then $(l, r) = (0, \min(p_1, p_2))$. Clearly, f'_k is an affine function with slope $2w$ on this interval.
- the values $f_k(l)$, $f_k(r)$, $f'_k(l)$, $f'_k(r)$ and $f'(T_k)$. As f' may be discontinuous at T_k, the latest value correspond to the derivative from the left.

With these informations for $f_1, \ldots, f_{k-1}$, we show we can calculate $\bar{s}_k$ and T_k and evaluate $f_k(x)$ or $f'_k(x)$ for any x in $O(k)$ time. We only illustrate how to solve the equation

$$f'_k(s) = 2w_k s - w_k P_{k-1}$$

which gives $\bar{s}_k$. The other procedures to compute T_k, $f_k(x)$ or $f'_k(x)$ are indeed very similar.

For $k > 1$, the value $\bar{s}_k$ is computed by calling the recursive procedure $\texttt{sbar}(k-1, w_k, w_k P_{k-1})$. Algorithm 4 $\texttt{sbar}(k, \omega, \pi)$ solves the equation $f'_k(s) = 2\omega s - \pi$ when $f''(s) > 2\omega$. It first checks whether the solution is T_k. Otherwise, using the fact that $s \mapsto f'_k(s) - 2\omega s$ is nondecreasing, it finds whether s is in $[0, l_k)$, $[l_k, r_k)$ or $[r_k, T_k)$ and accordingly studies one of the three cases:

- if $s \in [r_k, T_k)$, then according to equations (27) and (30) we have $f'_k(s) = f'_{k-1}(s) - w_k p_k$ so that we must have $f'_{k-1}(s) = 2\omega s - \pi + w_k p_k$.
- if $s \in [l_k, r_k)$, then f'_k is affine over the interval and $f'_k(s) = f'_k(l_k) + 2w_k(s - l_k)$. So s is solution of the linear equation $f'_k(l_k) + 2w_k(s - l_k) = 2\omega s - \pi$.
- if $s \in [0, l_k)$, we must check whether f_k was built with equation (27) or equation (30):
  - if $T_k < P_k/2$, $f'_k(s) = f'_{k-1}(s - p_k) + w_k p_k$. After changing the variable, s is equal to $t + p_k$ where t is the solution of $f'_{k-1}(t) = 2\omega t + 2\omega p_k - \pi - w_k p_k$.
  - otherwise, $f'_k(s) = -f'_{k-1}(P_{k-1} - s) - w_k p_k$. So, we have $s = P_{k-1} - t$ with $f'_{k-1}(t) = 2\omega t - w_k p_k + \pi - 2\omega P_{k-1}$.

We observe that in the first and third cases, the problem is solved by recursively solving the same problem with size $k - 1$. In the second case (which is the only possible case when $k = 1$), the problem is immediately solved and the recurrence is stopped. Therefore, $\bar{s}_k$ is computed in $O(k)$ time.

The values l_k and r_k are derived from $\bar{s}_k$. In order to compute T_k, we first compute the solution $s^\star$ by calling $\texttt{sbar}(k-1, 0, w_k p_k)$, which takes $O(k)$ time. The evaluation of $f_k(l)$, $f_k(r)$, $f'_k(l)$, $f'_k(r)$ and $f'(T_k)$ can also be done by a similar recursive procedure in $O(k)$ time. As a result, f_k can be derived from f_{k-1} in $O(k)$ time and we finally have the theorem.

Algorithm 4 Algorithm $\texttt{sbar}(k, \omega, \pi)$ to compute $\bar{s}$

if $f'_k(T_k) \leq 2\omega T_k - \pi$ **then**
  return T_k
else if $f'_k(r_k) \leq 2\omega r_k - \pi$ **then**
  return $\texttt{sbar}(k-1, \omega, \pi - w_k p_k)$
else if $f'_k(l_k) \leq 2\omega l_k - \pi$ **then**
  return $\frac{f'_k(l_k) - 2w_k l_k + \pi}{2(\omega - w_k)}$
else if $T_k < P_k/2$ **then**
  return $p_k + \texttt{sbar}(k-1, \omega, \pi + (w_k - 2\omega)p_k)$
else
  return $P_{k-1} - \texttt{sbar}(k-1, \omega, w_k p_k + 2\omega P_{k-1} - \pi)$
end if

Theorem 7 *The two-machine problem with linear costs can be solved in $O(n^2)$ time.*

We noted in Section 4.1. that the schedule has at most $n - 2$ interruptions since the first and last jobs can be scheduled without interruption. However this result can be improved by the following lemma, in which S_k is the start time of J_k on the second machine and C_k is its completion time (on the first machine).

Lemma 8 *In an optimal schedule, if $C_k > S_k + p_k$ then $C_{k+1} = S_{k+1} + p_{k+1}$.*

PROOF. Let us consider an optimal schedule in which we have both $C_k > S_k + p_k$ and $C_{k+1} > S_{k+1} + p_{k+1}$. Let x_k be the part of J_k on the second machine. We modify the schedule such that x_k is increased by ϵ and x_{k+1} is decreased by ϵ. As long as $0 \leq \epsilon \leq \delta$ with $\delta = \min(x_{k+1}, p_k - x_k, C_k - S_k - p_k) > 0$, the new schedule is feasible. Moreover, since $\alpha_{k+1} < \alpha_k$, its cost is strictly decreased, which contradicts the initial assumption that we can have $C_k > S_k + p_k$ and $C_{k+1} > S_{k+1} + p_{k+1}$ in an optimal schedule.

Therefore, there are at most $\lfloor (n-1)/2 \rfloor$ jobs such that $C_k > S_k + p_k$. Let us consider a job J_k with $C_k = S_k + p_k$ and let t be the time when J_k is stopped on machine 2 and starts on machine 1. From Lemma 6, a job completes at t on machine 1 and another one starts on machine 2. Therefore, by swapping all the jobs scheduled after t between machine 1 and machine 2, the interruption of J_k disappears and no new preemption appears. By iterating this transformation, a schedule with at most $\lfloor (n-1)/2 \rfloor$ interruptions is finally built.

5. Conclusion

In this paper, combinatorial algorithms have been proposed to efficiently solve three preemptive schedul-

ing problems with position costs. This new criterion is fundamentally different from classical criteria of the scheduling theory which are based on the completion times of the jobs. Consequently, the algorithms to solve these problems are not immediate adaptations of existing scheduling algorithms even if they are based on well-known Operations Research concepts (primal-dual approach and dynamic programming). The two main algorithms presented in this paper use some techniques that could be re-used to solve other problems: the auxiliary parameterized problem to solve the one-machine common due date problem and the compact encoding of the cost functions in the dynamic programming algorithm.

Other scheduling problems with positions costs are still to be investigated, in particular, problems with release dates and/or deadlines. The generalization of the algorithm presented in Section 3.2. is also interesting. For example, the same technique could be used to solve the problem with general due dates, however the analysis of the algorithm should be far more difficult and the number of events may not be polynomial. The algorithm could also be used to compute a lower bound for the non-preemptive common due date problem using the lower bounding scheme presented by Sourd [16] even if the relaxed problem to solve slightly differs from the problem studied in Section 3.2. because cost functions are not continuous anymore.

Acknowledgment

The author would like to thank Philippe Baptiste for early discussions about this work.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network flows: Theory, algorithms, and applications*, Prentice Hall, Upper Saddle River, NJ, 1993.

[2] R.K. Ahuja, J.B. Orlin, C. Stein, and R.E. Tarjan, *Improved algorithms for bipartite network flows*, SIAM Journal on Computation **23** (1994), 906–933.

[3] N. Brahimi, S. Dauzère-Pérès, N.M. Najid, and A. Nordli, *Single item lot sizing problems*, European Journal of Operational Research **168** (2006), 1–16.

Received 28 April 2005; revised 5 December 2005; accepted 22 April 2006

[4] N. Brauner, Y. Crama, A. Grigoriev, and van de Klundert, *A framework for the complexity of high-multiplicity scheduling problems*, Journal of Combinatorial Optimization **9** (2005), 313–323.

[5] P. Brucker, *Scheduling algorithms*, fourth edition ed., Springer-Verlag, Berlin, Germany, 2004.

[6] K. Bülbül, P. Kaminsky, and C. Yano, *Preemption in single machine earliness/tardiness scheduling*, Working paper, 2004.

[7] J.J. Clifford and M.E. Posner, *High multiplicity in earliness-tardiness scheduling*, Operations Research **48** (2000), 788–800.

[8] R. Fourer and R.E. Marsten, *Solving piecewise-linear programs: Experiments with a simplex approach*, ORSA Journal on Computing **4** (1992), 16–31.

[9] L. Gelders and P. Kleindorfer, *Coordinating aggregate and detailed scheduling decisions in the one-machine job shop. I. Theory*, Operations Research **22** (1974), 46–60.

[10] V. Gordon, J.M. Proth, and C. Chu, *A survey of the state-of-the-art of common due date assignment and scheduling research*, European Journal of Operational Research **139** (2002), 1–25.

[11] S. Kedad-Sidhoum, Y. Rios-Solis, and F. Sourd, *Lower bounds for the earliness-tardiness scheduling problem on single and parallel machines*, Working paper – `www.optimization-online.org`, October 2004.

[12] W. Kubiak, *Balancing mixed-model supply chain*, Graph Theory and Combinatorial Optimization (D. Avis, A. Hertz, and O. Marcotte, eds.), Gerad 25th Anniversary Series, Springer, 2005, pp. 159–190.

[13] V. Lauff and F. Werner, *Scheduling with common due date, earliness and tardiness penalties for multimachine problems: A survey*, Mathematical and Computer Modelling **40** (2004), 637–655.

[14] J. Y-T. Leung (ed.), *Handbook of scheduling: Algorithms, models and performance analysis*, Computer and Information Science Series, Chapman & Hall / CRC, Boca Raton, Florida, 2004.

[15] N. Rizk and A. Martel, *Supply chain flow planning methods: A review of the lot-sizing literature*, Tech. report, CENTOR Working Paper, 2001.

[16] F. Sourd, *The continuous assignment problem and its application to preemptive and non-preemptive scheduling with irregular cost functions*, INFORMS Journal on Computing **16** (2004), 198–208.

[17] ———, *Optimal timing of a sequence of tasks with general completion costs*, European Journal of Operational Research **165** (2005), 82–96.

[18] F. Sourd and S. Kedad-Sidhoum, *The one machine problem with earliness and tardiness penalties*, Journal of Scheduling **6** (2003), 533–549.